알고리즘과 파이썬 정오표

2021. 09. 16.

- 37페이지 연습문제 1.8

  $C_N = 2C_{N/2} + N^2,\ N \geq 2$이고 $C_1 = 1$.

- 40페이지

  아래에서 11줄 : ~ 예를 들어, 학생 리스트를 성명순으로 정렬할 때 이름이 같은 학생들의 상대적인 순서가 정렬 후에도 그대로 유지되면 안정적인 정렬 알고리즘이라고 할 수 있다.

- 47페이지 알고리즘 2.2

  ```
  bubbleSort(a[], n)
      for (i ← n; i > 1; i ← i-1) do {
  ```

- 49페이지 프로그램 2.2

  ```
  def bubbleSort(a, n):
      for i in range(n, 1, -1):
  ```

- 53페이지 알고리즘 2.3

  아래에서 3줄 : a[j] ← v;

- 59페이지 알고리즘 2.4

  위에서 2줄 : **for** (h ← 1; 3×h+1 < n; h ← 3×h+1) **do** { }; // 첫 번째 h 값 계산

- 60페이지 알고리즘 2.4

  아래에서 4줄 : a[j] ← v;

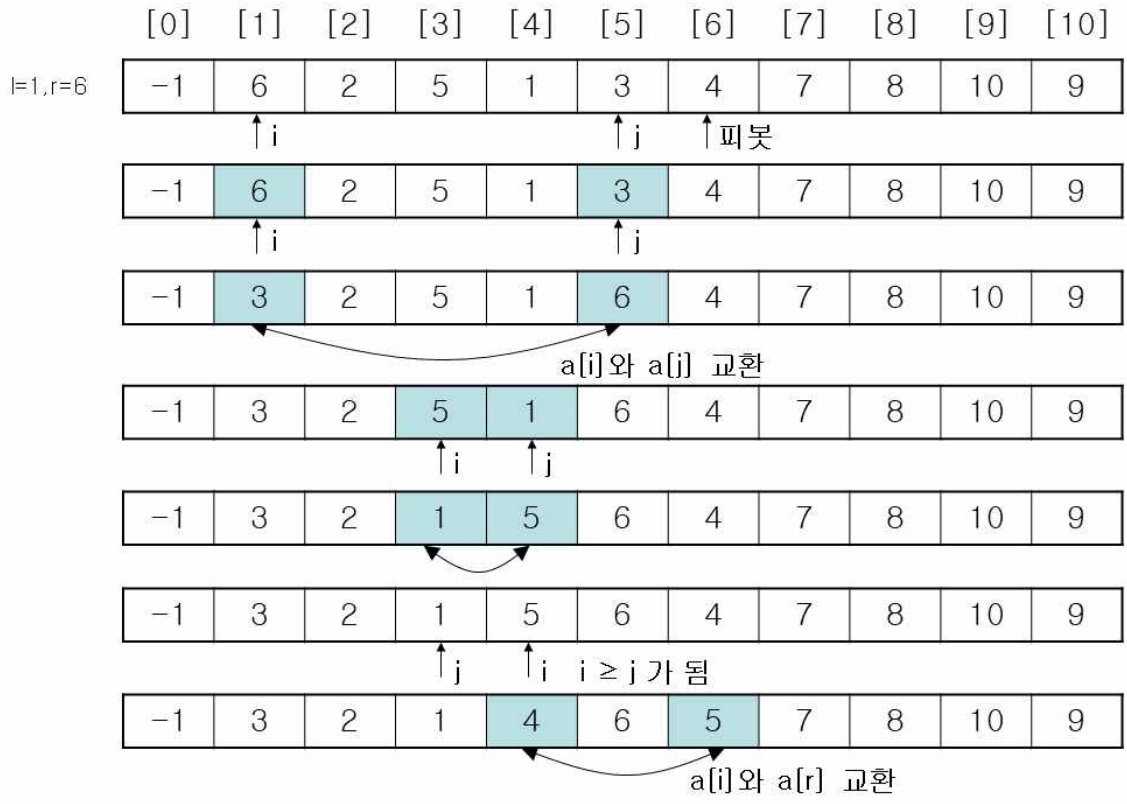
- 60페이지 [성능특성 2.4]

  [성능특성 2.4] 쉘 정렬 알고리즘의 성능은 인덱스 간격 순차에 따라 달라지는데, 최선의 경우 시간 복잡도는 $O(N\log N)$이고, 평균적인 경우는 $O(N^{4/3})$, 최악의 경우는 $O(N^{3/2})$으로 알려져 있다.

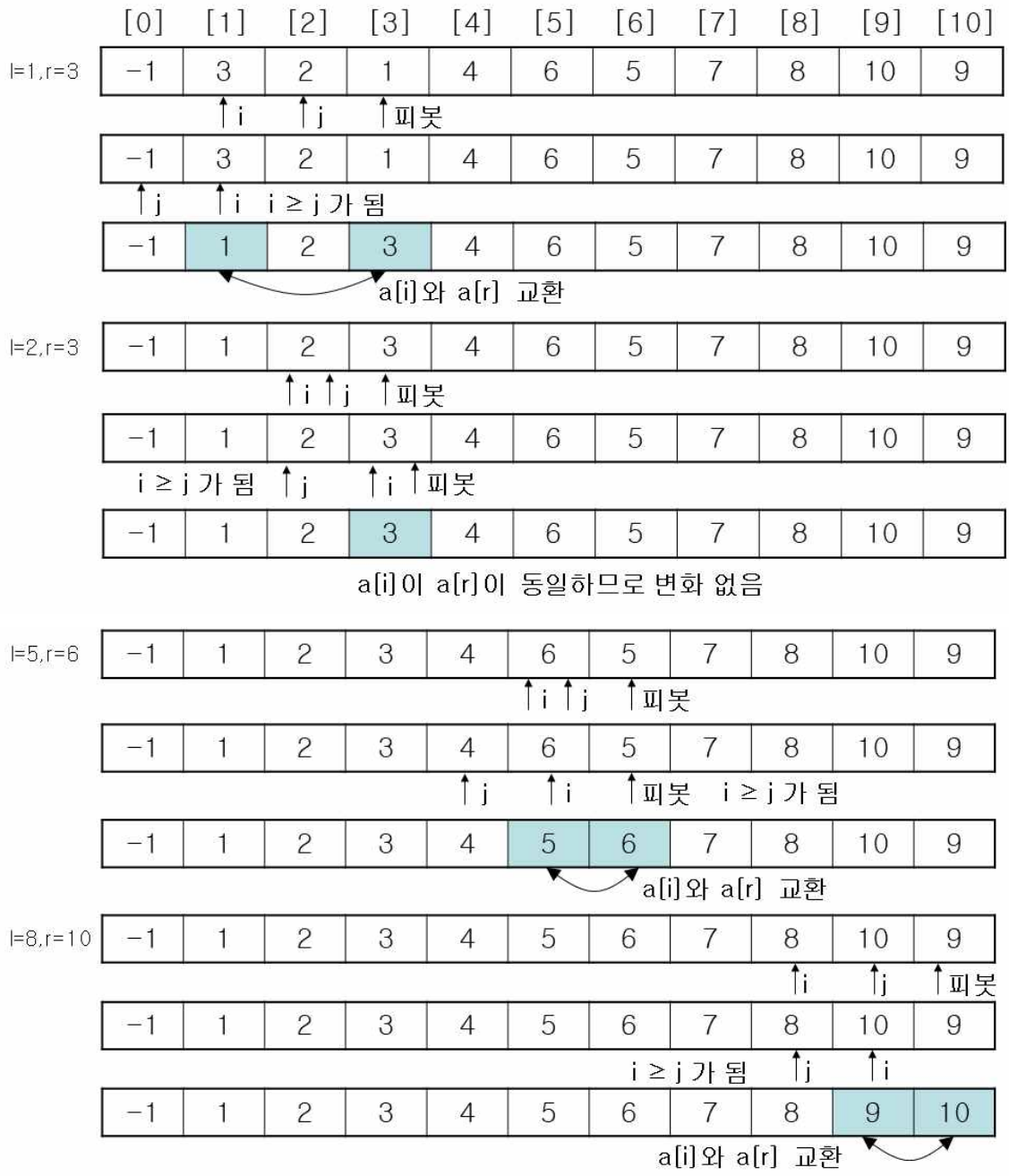
- 61페이지 프로그램 2.4

  위에서 3줄 : while 3 * h + 1 < n:

- 67페이지 그림 2.12



- 67페이지 그림 2.13



- 72페이지 프로그램 2.6

  위에서 13줄 : N = 100000

- 75페이지 알고리즘 2.7

  위에서 3줄 : m ← (l+r)/2;

- 77페이지 알고리즘 2.8

  위에서 1줄 : merge(a[], l, m, r)

- 79페이지 중간에 있는 소스코드

```
def mergeSort(a, l, r):
    if r > l:
        m = int((l+r)/2)
        mergeSort(a, l, m)
        mergeSort(a, m+1, r)
        i, j, k = l, m+1, l
        while i <= m and j <= r:
            if a[i] < a[j]:
                b[k] = a[i]
                k += 1
                i += 1
            else:
                b[k] = a[j]
                k += 1
                j += 1
        if i > m:
            for p in range(j, r+1):
                b[k] = a[p]
                k += 1
        else:
            for p in range(i, m+1):
                b[k] = a[p]
                k += 1
        for p in range(l, r+1):
            a[p] = b[p]
```

- 79페이지 프로그램 2.8

```
def mergeSort(a, l, r):
    if r > l:
        m = int((l+r)/2)
        mergeSort(a, l, m)
        mergeSort(a, m+1, r)
        i, j, k = l, m+1, l
        while i <= m and j <= r:
            if a[i] < a[j]:
                b[k] = a[i]
                k += 1
                i += 1
            else:
                b[k] = a[j]
                k += 1
                j += 1
        if i > m:
            for p in range(j, r+1):
                b[k] = a[p]
                k += 1
        else:
            for p in range(i, m+1):
                b[k] = a[p]
                k += 1
        for p in range(l, r+1):
            a[p] = b[p]

def checkSort(a, n):
    Sorted = True
    for i in range(1, n):
        if (a[i] > a[i+1]):
            Sorted = False
        if (Sorted == False):
            break
    if Sorted:
        print("정렬 완료₩n")
    else:
        print("정렬 오류 발생₩n")

import random, time
```

```
a = []
N = 100000
a.append(None)
for i in range(N):
    a.append(random.randint(1, N))
b = a.copy()
start_time = time.time()
mergeSort(a, 1, N)
end_time = time.time() - start_time
print('합병 정렬의 실행 시간 (N = %d) : %0.3f'%(N, end_time))
checkSort(a, N)
```

- 84페이지 알고리즘 2.9

```
heapSort(a[], n)
  for (i ← n/2; i ≥ 1; i ← i-1) do        // 배열 a[]를 히프로 변환
    heapify(a, i, n);                     // i는 내부 노드
  for (i ← n-1; i ≥ 1; i ← i-1) do {      // 배열 a[]를 오름차순으로 정렬
    a[1]과 a[i+1]을 교환;                 // a[1]은 제일 큰 원소
    heapify(a, 1, i);
  }
end heapSort()
```

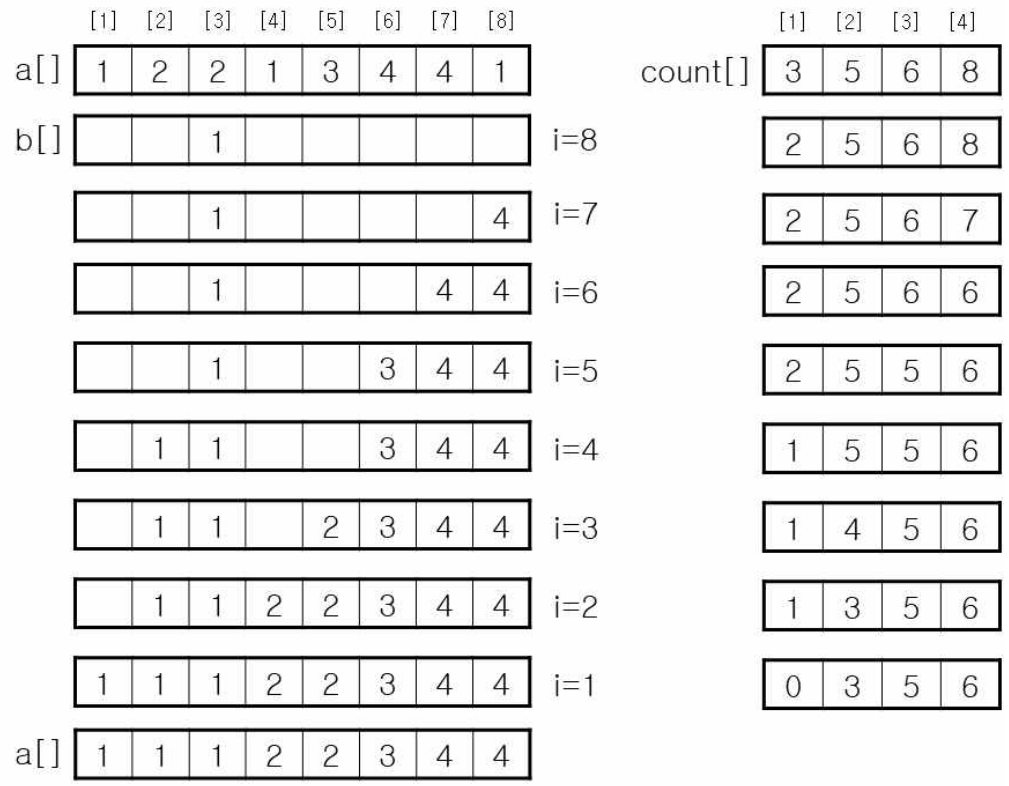
- 85페이지 알고리즘 2.10

  위에서 5줄 : **for** (j ← 2×h; j ≥ m; j ← 2×j) **do** {

- 91페이지 알고리즘 2.11

```
countingSort(a[], n, m)
  for (j ← 1; j ≤ m; j ← j + 1) do
    count[j] ← 0;          // count[]를 0으로 초기화
  for (i ← 1; i ≤ n; i ← i + 1) do
    count[a[i]] ← count[a[i]] + 1; // 원소의 개수를 세어 count[]에 저장
  for (j ← 2; j ≤ m; j ← j + 1) do
    count[j] ← count[j-1] + count[j]; // 원소가 들어 갈 위치 계산
  for (i ← n; i ≥ 1; i ← i - 1) do {
    b[count[a[i]]] ← a[i]; // a[]의 원소를 b[]의 미리 계산된 위치로 이동
    count[a[i]] ← count[a[i]] - 1; // count[]의 값을 하나 감소시킴
  }
  for (i ← 1; i ≤ n; i ← i + 1) do
    a[i] ← b[i];  // b[]를 a[]로 이동
end countingSort()
```

- 92페이지 그림 2.20



- 95페이지 알고리즘 2.12

  위에서 1줄 : radixSort(a[], n, m, Q)

  위에서 4줄 : **for** (i ← 1; i ≤ n; i ← i + 1) **do {**

- 110페이지 연습문제 2.10

  ㉡ [2, 7, 8, 11, 15, 5, 3, 1, 6, 13, 14, 9, 12, 10, 4]

  ㉢ [11, 8, 10, 6, 7, 9, 5, 1, 2, 4, 3, 12, 13, 14, 15]

- 116페이지 알고리즘 3.1

```
sequentialSearch(a[], search_key, n)
   i ← 0;
  while (i < n and a[i].key ≠ search_key) do i ← i + 1;
  if (i = n) then return -1   // key 값이 존재하지 않음
  else return i;
end sequentialSearch()
```

- 118페이지 프로그램 3.1

  아래에서 3줄 : start_time = time.time()

  아래에서 8줄 : end_time = time.time() - start_time

- 119페이지 알고리즘 3.2

  위에서 6줄 : **if** (a[mid].key > search_key) **then** right ← mid - 1;

- 121페이지 프로그램 3.2

  위에서 4줄 : start_time = time.time()

  위에서 9줄 : end_time = time.time() - start_time

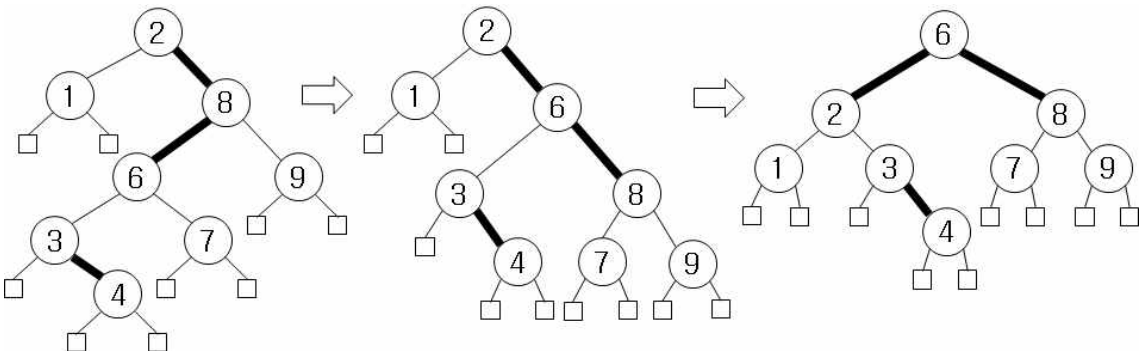
- 122페이지 알고리즘 3.3

```
binaryTreeSearch(T, search_key)
  x ← T;
  while (x ≠ null) do {
    if (x.key = search_key) then return x.key;        // 탐색 성공
    if (x.key > search_key) then x ← x.left;
        // 탐색 키가 작으므로 왼쪽 서브트리 탐색
    else x ← x.right;
        // 탐색 키가 크므로 오른쪽 서브트리 탐색
  }
  return –1;       // 탐색 실패
end binaryTreeSearch()
```

- 136페이지

  위에서 5줄 : ~. 이 경우는 [그림 3.11]과 같이 ~

- 139페이지 그림 3.14 (h)



- 140페이지 프로그램 3.4

```
black = 0
red = 1

class node:
    def __init__(self, color, key=None, left=None, right=None):
```

```
        self.color = color
        self.key = key
        self.left = left
        self.right = right

class Dict:
    z = node(color=black, key=0, left=0, right=0)
    z.left = z
    z.right = z
    head = node(color=black, key=0, left=0, right=z)

    def search(self, search_key):
        x = self.head.right
        while (x != self.z):
            if x.key == search_key:
                return x.key
            if x.key > search_key:
                x = x.left
            else:
                x = x.right
        return -1

    def insert(self, v):
        x = p = g = self.head
        while (x != self.z):
            gg = g
            g = p
            p = x
            if x.key == v:
                return
            if x.key > v:
                x = x.left
            else:
                x = x.right
            if x.left.color and x.right.color:
                self.split(x, p, g, gg, v)
        x = node(color=red, key=v, left=self.z, right=self.z)
        if p.key > v:
            p.left = x
        else:
            p.right = x
```

```
            self.split(x, p, g, gg, v)
            self.head.right.color = black

        def split(self, x, p, g, gg, v):
            x.color = red
            x.left.color = black
            x.right.color = black
            if p.color:
                g.color = red
                if (g.key > v) != (p.key > v):
                    p = self.rotate(v, g)
                x = self.rotate(v, gg)
                x.color = black

        def rotate(self, v, y):
            if y.key > v:
                c = y.left
            else:
                c = y.right
            if c.key > v:
                gc = c.left
                c.left = gc.right
                gc.right = c
            else:
                gc = c.right
                c.right = gc.left
                gc.left = c
            if y.key > v:
                y.left = gc
            else:
                y.right = gc
            return gc

import random, time

N = 10000
key = list(range(1,N + 1))
s_key = list(range(1, N + 1))
random.shuffle(key)

d = Dict()
```

```
for i in range(0, N):
    d.insert(key[i])
start_time = time.time()
for i in range(N):
    result = d.search(s_key[i])
    if result == -1 or result != s_key[i]:
        print("탐색 오류")
end_time = time.time() - start_time
print('레드 블랙 트리 탐색의 실행 시간 (N = %d) : %0.3f'%(N, end_time))
print('탐색 완료')
```

- 151페이지 프로그램 3.5

  위에서 1줄 : start_time = time.time()

  위에서 6줄 : end_time = time.time() - start_time

- 156페이지

  아래에서 2줄 : … 키 [1, 19, 5, 1, 18, 3, 8, 9, 14, 7, 5, 24, 1, 13, 16, 12, 5] …

- 157페이지 그림 3.21

  아래에서 3줄

19	1	1	3	1	5	5	7	8	9	24			13	14		16		18

- 159페이지 프로그램 3.6

  아래에서 9줄 : start_time = time.time()

  아래에서 4줄 : end_time = time.time() - start_time

- 161페이지

  아래에서 4줄 : … 키 [1, 19, 5, 1, 18, 3, 8, 9, 14, 7, 5, 24, 1, 13, 16, 12, 5] …

- 164페이지 프로그램 3.7

  아래에서 4줄 : start_time = time.time()

- 165페이지 프로그램 3.7

  위에서 2줄 : end_time = time.time() - start_time

- 167페이지

  아래에서 2줄 : [1, 19, 5, 18, 3, 26, 9]

- 168페이지

  위에서 2줄 : …. [그림 3.24]의 …

  위에서 5줄 : …. [그림 3.24]의 …

  위에서 6줄 : …. [그림 3.24]의 …

- 174페이지

  위에서 6줄 : …. 예를 들어, [그림 3.25]의 …

- 171페이지 프로그램 3.8

  아래에서 5줄 : start_time = time.time()

- 172페이지 프로그램 3.8

  위에서 1줄 : end_time = time.time() - start_time

- 172페이지

  아래에서 2줄 : [1, 19, 5, 18, 3, 26, 9]

- 177페이지 프로그램 3.9

  아래에서 8줄 : start_time = time.time()

  아래에서 3줄 : end_time = time.time() - start_time

- 178페이지

  위에서 1줄 : … 디지털 탐색 트리의 단점

  아래에서 4줄 : [1, 19, 5, 18, 3, 26, 9]

- 180페이지

  위에서 13줄 : 예를 들어, [그림 3.26]의 …

- 183페이지 프로그램 3.10

  아래에서 8줄 : start_time = time.time()

  아래에서 3줄 : end_time = time.time() - start_time

- 197페이지 프로그램 4.1

```
def bruteForce(p, t, s):
    M = len(p)
    N = len(t)
    i = s
    j = 0
    while j < M and i < N:
        if t[i] != p[j]:
            i -= j
            j = -1
        i += 1
        j += 1
    if j == M: return i - M
    else: return i

text = 'abababababcababababcaabbabababababcaab' + '₩0'
pattern = 'abababca'
m = len(pattern)
n = len(text)
start = 0
while True:
    position = bruteForce(pattern, text, start)
    start = position + 1
    if start <= n - m:
        print('패턴이 나타난 위치 : ', position)
    else: break
print('스트링 탐색 종료')
```

- 198페이지

위에서 1줄 : 패턴이 나타난 위치 : 2

위에서 2줄 : 패턴이 나타난 위치 : 11

위에서 3줄 : 패턴이 나타난 위치 : 22

- 199페이지 알고리즘 4.2

  위에서 3줄 : initNext(p);

- 202페이지 알고리즘 4.3

  위에서 4줄 : for (i ← 0, j ← −1; i < M; i ← i + 1, j ← j + 1) do {

- 207페이지 프로그램 4.4

```
def initNext(p):
    M = len(p)
    next[0] = -1
    i = 0
    j = -1
    while i < M:
        if j != -1 and p[i] == p[j]: next[i] = next[j]
        else: next[i] = j
        while j >= 0 and p[i] != p[j]:
            j = next[j]
        i += 1
        j += 1

def KMP(p, t, s):
    M = len(p)
    N = len(t)
    initNext(p)
    i = s
    j = 0
    while j < M and i < N:
        while j >= 0 and t[i] != p[j]:
            j = next[j]
        i += 1
        j += 1
    if j == M: return i - M
    else: return i

next = [0] * 50
text = 'abababababcababababcaabbabababababcaab' + '₩0'
```

```
pattern = 'abababca'
m = len(pattern)
n = len(text)
start = 0
while True:
    position = KMP(pattern, text, start)
    start = position + 1
    if start <= n - m: print('패턴이 나타난 위치 : ', position)
    else: break
print('스트링 탐색 종료')
```

- 208페이지 중간

j	1	2	3	4	5	6	7
next[j]	0	−1	0	−1	0	4	−1

- 210페이지 알고리즘 4.5

  위에서 6줄 : s ← skip[index(t[i])];

  위에서 7줄 : if (M-j > s) then i ← i + M - j;

  위에서 8줄 : else i ← i + s;

- 211페이지

  아래에서 8줄 : ... i ← i + M − j ...

- 212페이지 프로그램 4.5

```
def index(c):
    if ord(c) == 32: return 0
    else: return ord(c)-64

def initSkip(p):
    M = len(p)
```

```
    for i in range(NUM):
        skip[i] = M
    for i in range(M):
        skip[index(p[i])] = M - i - 1

def BM(p, t, s):
    M = len(p)
    N = len(t) - 1
    initSkip(p)
    i = s + M - 1
    j = M - 1
    if i >= N: return N
    while j >= 0:
        while t[i] != p[j]:
            k = skip[index(t[i])]
            if M - j > k: i += M - j
            else: i += k
            if i >= N: return N
            j = M - 1
        i -= 1
        j -= 1
    return i + 1

NUM = 27
skip = [0] * NUM
text = 'VISION QUESTION ONION CAPTION GRADUATION EDUCATION' + '₩0'
pattern = 'ATION'
m = len(pattern)
n = len(text)
start = 0
while True:
    position = BM(pattern, text, start)
    start = position + 1
    if start <= n - m: print('패턴이 나타난 위치 : ', position)
    else: break
print('스트링 탐색 종료')
```

- 215페이지

  아래에서 9줄 : … dM = 10^4 mod 13 = 3 …

아래에서 6줄 : … ((31415 − 3 • 10^4)10 + 2) …

- 216페이지 알고리즘 4.6

위에서 4줄 : for (i ← 1; i < M; i ← i + 1) do

위에서 6줄 : for (i ← 0; i < M; i ← i + 1) do {

아래에서 5줄 : h2 ← (h2 * d + index(t[i+M])) mod q;

- 217페이지 프로그램 4.6

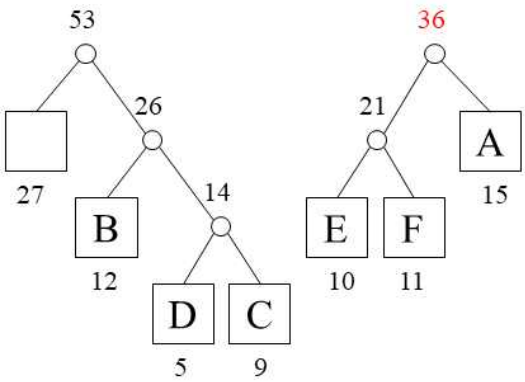
```
def index(c):
    if ord(c) == 32: return 0
    else: return ord(c)-64

def RK(p, t, s):
    dM = 1
    h1 = 0
    h2 = 0
    M = len(p)
    N = len(t)
    for i in range(1, M):
        dM = (d * dM) % q
    for i in range(M):
        h1 = (h1 * d + index(p[i])) % q
    i = s
    j = 0
    while i < N and j < M:
        h2 = (h2 * d + index(t[i])) % q
        i += 1
        j += 1
    i = s
    while h1 != h2:
        if i + M >= N: return N
        h2 = (h2 + d * q - index(t[i]) * dM) % q
        h2 = (h2 * d + index(t[i + M])) % q
        if i > N - M: return N
```

```
            i += 1
        return i

q = 33554393
d = 32
text = 'VISION QUESTION ONION CAPTION GRADUATION EDUCATION' + '₩0'
pattern = 'ATION'
m = len(pattern)
n = len(text)
start = 0
while True:
        position = RK(pattern, text, start)
        start = position + 1
        if start <= n - m: print('패턴이 나타난 위치 : ', position)
        else: break
print('스트링 탐색 종료')
```

- 232페이지 그림 4.19



(f) 26과 27 삭제

- 233페이지 표 4.3

		A	B	C	D	E	F
k	0	1	2	3	4	5	6
length[k]	2	2	3	4	4	3	3
code[k]	0	3	2	7	6	4	5
	00	11	010	0111	0110	100	101

- 233페이지

  표 4.3 바로 아래 : … length[k] …

- 235페이지 프로그램 4.8

  위에서 1줄 : i = 27

- 236페이지 프로그램 4.8

  위에서 4줄 : if cnt % 70 == 0:

- 245페이지

  아래에서 7줄 : … 약 10^6년

- 246페이지

  위에서 2줄 : … C = P(M) = M^P mod N …

  위에서 3줄 : … M = S(C) = C^S mod N …

- 251페이지 연습문제 4.6

  숫자로 변환된 텍스트 43875에 대해 q=13으로 라빈-카프 알고리즘을 수행한다고 할 때, 다음 물음에 답하라. 다음 계산 결과를 참고하라.

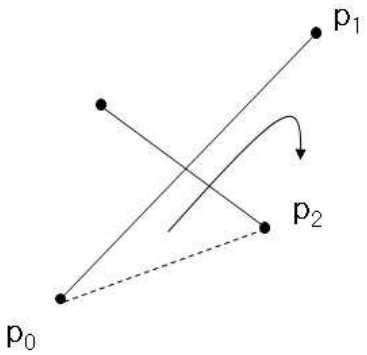
1000 mod 13 = 12

  (1) 4387 mod 13의 값을 호너의 방법으로 구하라.
  (2) (1)의 계산 결과를 사용하여 3875 mod 13의 값을 라빈-카프 알고리즘으로 구하라.

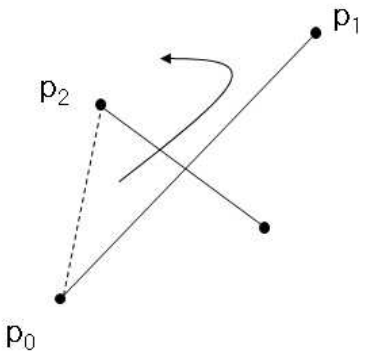
- 251페이지

  위에서 2줄 : 암호문을 복호화하라. …

- 258페이지 그림 5.3



(a) 시계 방향



(b) 시계 반대 방향

- 279페이지 알고리즘 5.4 아래에서 2줄 '}' 추가

```
    if (minIndex = n) then return m;
  }
end packageWrapping()
```

- 302페이지 알고리즘 5.7

```
rangeSearch(B, v1, v2)
  t ← B;
  count ← 0;
  if (t = null) then return 0;
  if (t.key ≥ v1) then
    count ← count + rangeSearch(t.left, v1, v2);
  if (t.key ≥ v1 and t.key ≤ v2) then
    count ← count + 1;
  if (t.key ≤ v2) then
    count ← count + rangeSearch(t.right, v1, v2);
  return count;
end rangeSearch()
```

- 314페이지 알고리즘 5.10 위에서 6줄부터

$tx_1 \leftarrow range.x_1 < t.p.x$; $tx_2 \leftarrow range.x_2 \geq t.p.x$;
$ty_1 \leftarrow range.y_1 < t.p.y$; $ty_2 \leftarrow range.y_2 \geq t.p.y$;
if (d = true) **then** $t_1 \leftarrow tx_1$ **else** $t_1 \leftarrow ty_1$;
if (d = true) **then** $t_2 \leftarrow tx_2$ **else** $t_2 \leftarrow ty_2$;

- 331페이지 연습문제 5.1

	ccw(p_0, p_1, p_2)
	$dx_1 \leftarrow p_1.x - p_0.x$; $dy_1 \leftarrow p_1.y - p_0.y$;
	$dx_2 \leftarrow p_2.x - p_0.x$; $dy_2 \leftarrow p_2.y - p_0.y$;
①	**if** ($dx_1.dy_2 > dy_1.dx_2$) **then return** +1;
②	**if** ($dx_1.dy_2 < dy_1.dx_2$) **then return** −1;
③	**if** (($dx_1 = 0$) and ($dy_1 = 0$)) **then return** 0;
④	**if** (($dx_1.dx_2 < 0$) or ($dy_1.dy_2 < 0$) **then return** −1;
⑤	**if** (($dx_1^2+dy_1^2$) < ($dx_2^2+dy_2^2$)) **then return** +1;
⑥	**return** 0;
	end ccw()

다음과 같이 ccw(p_0, p_1, p_2) 함수를 호출한다고 할 때, 알고리즘의 몇 번 문장이 실행되는가?

(1) p_0 = (5, 2), p_1 = (7, 5). p_2 = (11, 1) 일 때
(2) p_0 = (7, 5), p_1 = (13, 8). p_2 = (4, 15) 일 때
(3) p_0 = (5, 7), p_1 = (7, 10). p_2 = (9, 13) 일 때
(4) p_0 = (12, 9), p_1 = (9, 11). p_2 = (15, 7) 일 때
(5) p_0 = (6, 9), p_1 = (12, 13). p_2 = (9, 11) 일 때

- 341페이지

아래에서 11줄 : $2^2 t_{n-2}$ …

- 342페이지

위에서 2줄 : M [i, j] = $\min_{i \leq k \leq j-1}$ (…)

- 343페이지

  위에서 3줄 : M[1, 3] + M[4, 4] + $d_0d_3d_4$)

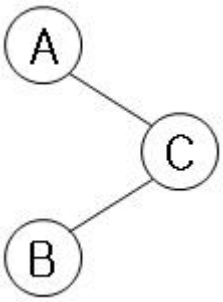
  위에서 8줄 : M[1, 3] + M[4, 5] + $d_0d_3d_5$, …)

  아래에서 7줄 : M[1, 3] + M[4, 6] + $d_0d_3d_6$, …)

- 348페이지

  아래에서 2줄 : (e) 1 × 0.6 + 2 × 0.3 + 3 × 0.1 = 1.5

- 349페이지 그림 6.4(a)



(a)

- 350페이지

  위에서 4줄 : A[i, j] = $\min_{i \le k \le j}$ (…)

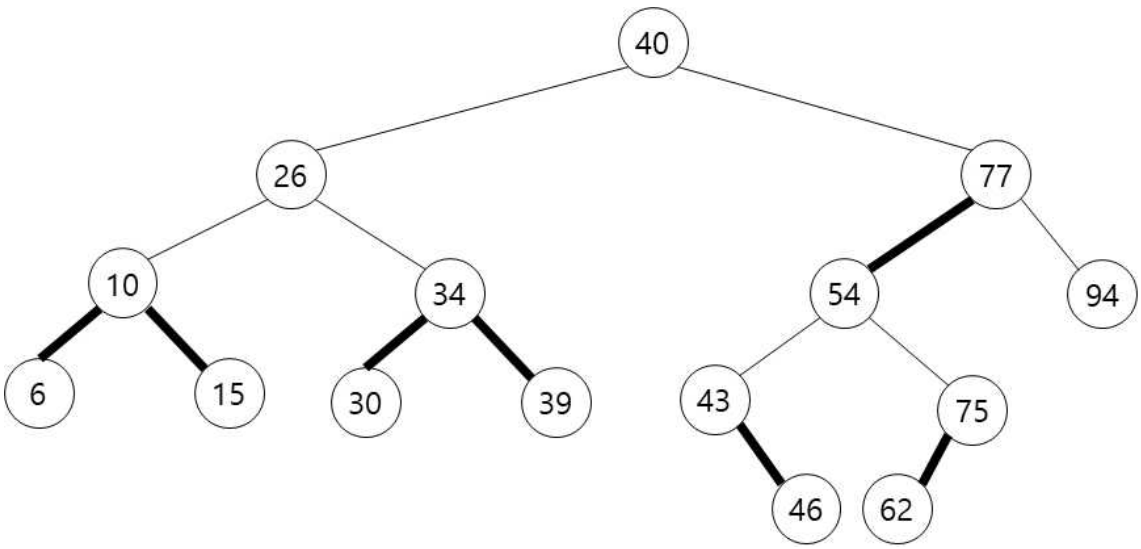
  위에서 5줄 : = $\min_{i \le k \le j}$ (…)

- 352페이지 표 6.3(b)

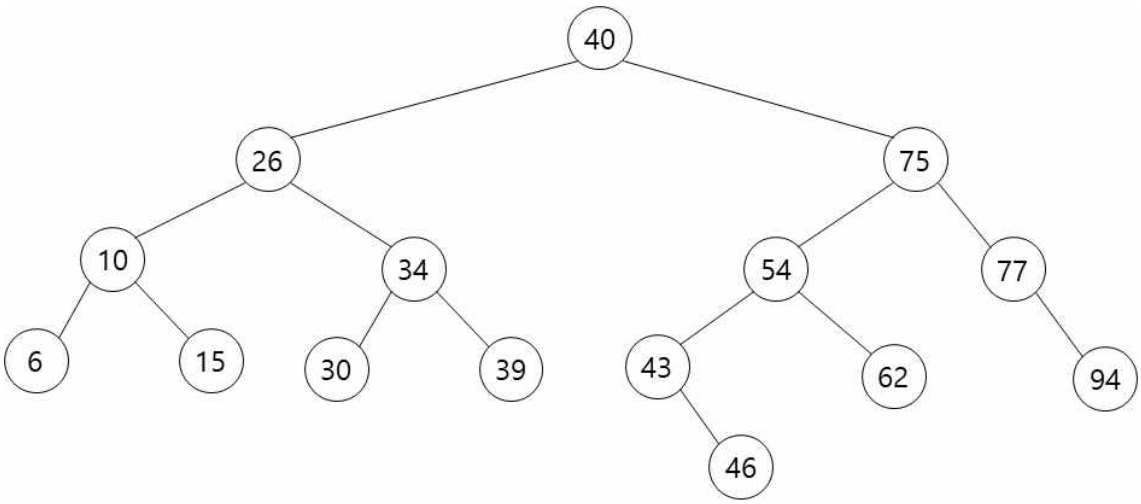
i \ j	1	2	3	4
1	1	1	2	3
2	0	2	3	3
3	0	0	3	3
4	0	0	0	4

- 381페이지

3.1 (2)

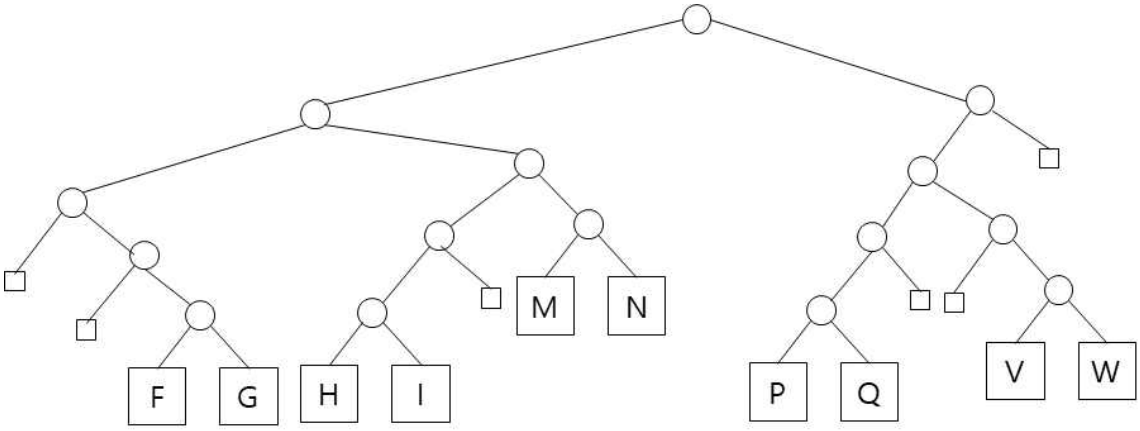


3.1 (3)



- 383페이지

3.4 (2)



- 385페이지

4.6 (1)

4387 mod 13
= $(4 \cdot 10^3 + 3 \cdot 10^2 + 8 \cdot 10^1 + 7 \cdot 10^0)$ mod 13
= (7 + 10(8 + 10(3 +10 • 4)) mod 13
= (7 + 10(8 + 10(43 mod 13))) mod 13
= (7 + 10(8 + 10 • 4)) mod 13
= (7 + 10(48 mod 13)) mod 13
= (7 + 10 • 9) mod 13
= 97 mod 13
= 6

4.6 (2)

3875 mod 13 = $((4387 - 4 \cdot 10^3)10 + 5)$ mod 13
= ((6 + 10 • 13 - 4 • 12)10 + 5) mod 13
= ((88 mod 13)10 + 5) mod 13
= (10 • 10 + 5) mod 13
= 105 mod 13
= 1

- 386페이지

4.8 (2)

k	0	1	2	3	4	5	6	7	9	12	13	14	15	16	18	19	20	21
count[k]	11	3	3	1	2	5	1	2	6	2	4	5	3	1	2	4	3	2
dad[k]	-40	-32	-33	27	-28	36	-27	29	-37	-30	35	37	33	28	31	-35	32	30

k	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
count[k]	2	3	4	4	5	6	6	8	8	10	11	12	16	21	23	37	60
dad[k]	-29	-31	-34	34	-36	-38	38	39	-39	40	41	-41	42	-42	43	-43	0

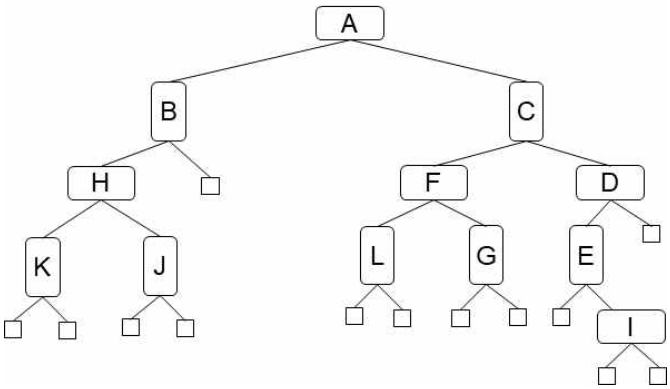
- 387페이지

5.4

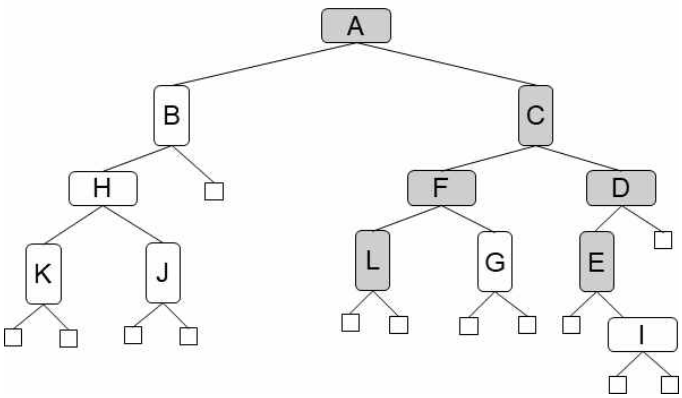
minValue	cp1	cp2	p1	p2	p3	p4
1000						H
$\sqrt{17}$	D	H			H	D
$\sqrt{17}$	D	H				A
$\sqrt{17}$	D	H			A	C
$\sqrt{17}$	D	H				H
$\sqrt{10}$	A	H			H	A
$\sqrt{10}$	A	H		H	A	D
$\sqrt{10}$	A	H	H	A	D	C
$\sqrt{10}$	A	H				E
$\sqrt{10}$	A	H			E	B
$\sqrt{10}$	A	H				G
$\sqrt{5}$	F	G			G	F
$\sqrt{5}$	F	G				G
$\sqrt{5}$	F	G			G	F
$\sqrt{5}$	F	G		G	F	E
$\sqrt{5}$	F	G				A
$\sqrt{5}$	F	G			A	E
$\sqrt{5}$	F	G		A	E	B

- 388페이지

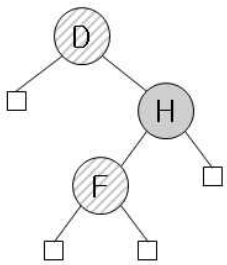
5.5 (1)



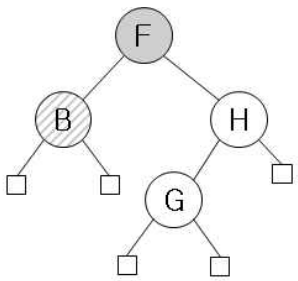
5.5 (2)



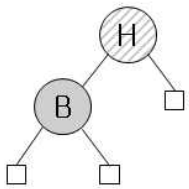
5.6 (3)



D, H, F 삽입 후
수평선 C를 만났을 때



G, B 삽입, D 삭제 후
수평선 A를 만났을 때



G, F 삭제 후
수평선 E를 만났을 때

- 389페이지

6.1 (1)

M [i, j] = min $_{i \le k \le j-1}$ (…)

6.2 (2)

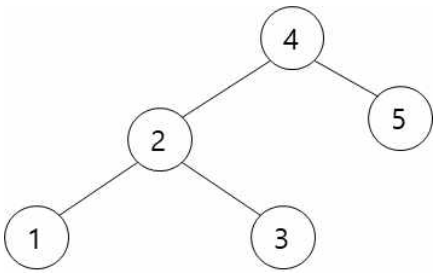
i \ j	1	2	3	4	5
1	0.21	0.43	0.85	1.49	2.08
2	0	0.11	0.38	0.94	1.40
3	0	0	0.16	0.61	1.07
4	0	0	0	0.29	0.75
5	0	0	0	0	0.23

$A[i,j]$의 값

i \ j	1	2	3	4	5
1	1	1	2	3	4
2	0	2	3	4	4
3	0	0	3	4	4
4	0	0	0	4	4
5	0	0	0	0	5

최소 값을 갖는 k의 값

6.2 (3)



- 390페이지

6.4 (1)

$$D[i,j] = \min(D[i,j-1]+1, D[i-1,j]+1, D[i-1,j-1]+0/2)$$